29｜周边：框架发布和维护也是重要的一环

2021-11-22 叶剑峰

《手把手带你写一个Web框架》 课程介绍 >

讲述：叶剑峰

时长 15:50 大小 14.51M

你好，我是轩脉刃。

终于来到框架设计与完善的最后一节课了。在前面的章节中，我们基本上把框架的功能都开发完成了，但是这只是万里长征的第一步。一个工业级的 Web 框架一定是经过长时间千锤百炼的迭代升级的。在这门课编写完成的时候，我为 hade 框架锁定了 v 1.0.0 版本，后续我们会继续为框架增加更多的功能和特性。

那么随着框架的不断更新和升级，随之而来的问题就是如何为一个开源项目设计一套发布和使用机制，并且为每个发布版本维护一套准确的框架说明文档？这就是我们今天要讲的内容。

版本

每个框架发布都需要有一个版本号，这个版本号如何定义，我们在前面的课程中已经不止一次提到过了，这里再正式说明一下。

所有开源软件的版本号基本上都遵循“语义化版本规范”的约定，这份语义化版本规范是由 Gravatars 创办者兼 GitHub 共同创办者 Tom Preston-Werner 所建立的，它定义了三段式的版本规范，格式如下：

```
1 主版本号.次版本号.修订号
```

复制代码

我们使用的 Golang 语言项目也是基于这个规范来实现的。

主版本号代表如果做了不兼容的 API 修改。比如在你的项目中，原先提供的 A 方法要替换为 B 方法，所有的参数和返回值都已经变化了，你的使用方必须修改他的代码，这个变动就叫做“不兼容的 API 修改”。这个时候，主版本号就必须更新了。

次版本号表示当你做了向下兼容的功能性新增。比如原先你的类中只有 A 方法，当你新增了一个 B 方法，但是并不修改原先的 A 方法，这个时候你的类有 A、B 两个方法。原先的库使用者并不需要更新他的任何代码，这就是“向下兼容的功能性新增”。这个时候，你不需要更新主版本号，只需要更新次版本号就行。

修改号，表示当你做了向下兼容的问题修正。比如原先类中只有 A 方法，你并没有更新任何的功能，只是修改了 A 方法中的一个 bug，那么这个时候，不需要更新主版本号和次版本号，直接更新修订版本号就行。

这个语义化版本规范，基本上已经是开源届的共识了，当然也可以不遵循这个规范，但是一旦不遵循，对你开源项目的使用者来说绝对是一个灾难性的事情，进而你的项目也逐渐会失去使用者。

我们使用的 Go 语言版本就是严格遵循这个规范的。比如现在使用的 Golang 版本为 1.17.2。主版本为 1，次版本为 17，修改号为 2。基本上我们接触 Golang 都是从 1.0.0 之后开始的，目前 Go 还没发布 2.0.0，所以 1.0 之后的 Go 版本升级都是向下兼容的。如果有做过 Go 升级的同学可以思考下，在升级的时候，业务代码有没有做任何变动。

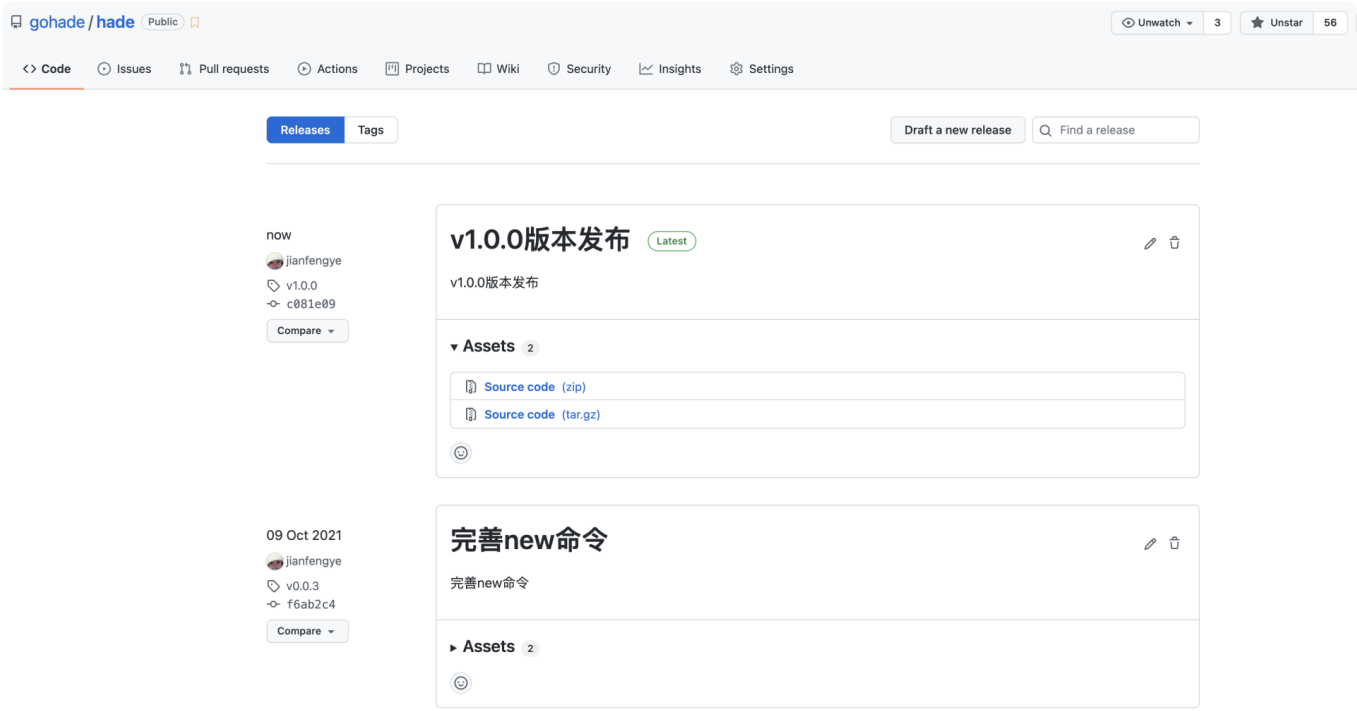
另外再附带说一下，你也有可能在网络上看到版本号不止三个的，比如 1.2.3.4，或者 1.2.3.beta。这些都是对这个语义化规范的扩展使用而已，最后一个字段根据不同的项目可能有不同的意思，比如是否是发行版本。不过前三个版本号都是遵循这个版本规范的。所以见到这些扩展版本号不要吃惊。

我们的 hade 框架当然也要基于这个版本规范，在这门课程更新完成之后，我会把这门课程的所有代码锁定为一个发布版本 1.0.0，并且从这个版本开始继续迭代更新功能。有新的功能和模块增加，我们会升级次版本号，有任何的 bug 修复，就升级修订版本号。

发布

明确了 hade 的版本号规范之后，要明确我们的发布模式，这个在前面的第 22 节课开发自动化脚手架已经简单说过了，这里再正式说明一下。

我们的框架最终在 GitHub 上的地址为：https://github.com/gohade/hade ，会至少为每一个次版本号打上 release 版本。



而对于框架的使用者来说，使用步骤会是这样的。首先需要在工作机器上安装 hade 命令，我们可以在任何路径调用：

```
1 go install github.com/gohade/hade@latest
```

来安装 hade 命令。这个 hade 命令，其实用任何版本都是可以的，这里直接选用最新版本。

调用了 go install 之后，hade 命令就被安装到你的 $GOPATH/bin 目录下了。直接调用 `$GOPATH/bin/hade` 可以看到安装好的 hade 命令。

```
~/Documents/gopath/bin $GOPATH/bin/hade
hade 框架提供的命令行工具，使用这个命令行工具能很方便执行框架自带命令，也能很方便编写业务命令

Usage:
  hade [flags]
  hade [command]

Available Commands:
  app         业务应用控制命令
  build       编译相关命令
  command     控制台命令相关
  config      获取配置相关信息
  cron        定时任务相关命令
  dev         调试模式
  env         获取当前的App环境
  foo         foo
  go          运行path/go程序，要求go 必须安装
  help        Help about any command
  middleware  中间件相关命令
  new         创建一个新的应用
  npm         运行 PATH/npm 的命令
  provider    服务提供相关命令

Flags:
  -h, --help   help for hade
```

然后使用 `hade new` 命令创建一个项目，比如为 hellohade。我们可以看到创建出了目标文件夹 hellohade。

```
~/Documents/workspace $GOPATH/bin/hade new
? 请输入目录名称： hellohade
? 请输入模块名称(go.mod中的module, 默认为文件夹名称):
? 请输入版本名称(参考 https://github.com/gohade/hade/releases, 默认为最新版本): v1.0.0
=================================================
开始进行创建应用操作
创建目录： /Users/yejianfeng/Documents/workspace/hellohade
应用名称： hellohade
hade框架版本： v1.0.0
创建临时目录 /Users/yejianfeng/Documents/workspace/template-hade-v1.0.0-1636680469
下载zip包到template.zip
解压zip包
删除临时文件夹 /Users/yejianfeng/Documents/workspace/template-hade-v1.0.0-1636680469
删除.git目录
删除framework目录
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/app/console/kernel.go
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/app/http/module/demo/api.go
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/app/http/module/demo/mapper.go
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/app/http/route.go
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/app/http/swagger.go
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/app/provider/kernel.go
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/docs/guide/app.md
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/go.mod
更新文件:/Users/yejianfeng/Documents/workspace/hellohade/main.go
创建应用结束
目录： /Users/yejianfeng/Documents/workspace/hellohade
=================================================
```

进入目标文件夹 `cd hellohade` ，调用 `go mod tidy` 下载所有的依赖包，调用 `go build` 编译目录。

```
~/Documents/workspace/hellohade go build
~/Documents/workspace/hellohade ll
total 116208
-rw-r--r--   1 yejianfeng  staff   1.1K Nov 12 09:27 LICENSE
-rw-r--r--   1 yejianfeng  staff   501B Nov 12 09:27 README.md
drwxr-xr-x   5 yejianfeng  staff   160B Nov 12 09:27 app
drwxr-xr-x  10 yejianfeng  staff   320B Nov 12 09:27 build
drwxr-xr-x   9 yejianfeng  staff   288B Nov 12 09:27 config
drwxr-xr-x   6 yejianfeng  staff   192B Nov 12 09:27 docs
-rw-r--r--   1 yejianfeng  staff   173B Nov 12 09:29 go.mod
-rw-r--r--   1 yejianfeng  staff    94K Nov 12 09:29 go.sum
-rwxr-xr-x   1 yejianfeng  staff    56M Nov 12 09:29 hellohade
-rw-r--r--   1 yejianfeng  staff   266B Nov 12 09:27 index.html
-rw-r--r--   1 yejianfeng  staff   1.7K Nov 12 09:27 main.go
-rw-r--r--   1 yejianfeng  staff   945K Nov 12 09:27 package-lock.json
-rw-r--r--   1 yejianfeng  staff   3.1K Nov 12 09:27 package.json
drwxr-xr-x   7 yejianfeng  staff   224B Nov 12 09:27 src
drwxr-xr-x   3 yejianfeng  staff    96B Nov 12 09:27 static
drwxr-xr-x   4 yejianfeng  staff   128B Nov 12 09:27 storage
drwxr-xr-x   5 yejianfeng  staff   160B Nov 12 09:27 test
drwxr-xr-x   3 yejianfeng  staff    96B Nov 12 09:27 tests
```

接下来，使用者就可以在这个目标文件夹中开始基于 hade 开发应用了。

```
~/Documents/workspace/hellohade ./hellohade
hade 框架提供的命令行工具，使用这个命令行工具能很方便执行框架自带命令，也能很方便编写业务命令

Usage:
  hade [flags]
  hade [command]

Available Commands:
  app         业务应用控制命令
  build       编译相关命令
  command     控制台命令相关
  config      获取配置相关信息
  cron        定时任务相关命令
  deploy      部署相关命令
  dev         调试模式
  env         获取当前的App环境
  foo         foo
  go          运行path/go程序，要求go 必须安装
  help        Help about any command
  middleware  中间件相关命令
  new         创建一个新的应用
  npm         运行 PATH/npm 的命令
  provider    服务提供相关命令
  swagger     swagger对应命令

Flags:
  -h, --help   help for hade

Use "hade [command] --help" for more information about a command.
```

文档维护

到这里，我们把框架的版本和发布的流程梳理清楚了。但是随着版本不断迭代更新，框架对应的说明文档也是需要不断更新的，这就涉及框架文档的编写和搭建了。

对于一个开源项目，说明文档的查看渠道是多种多样的。

对于比较小型的项目和类库，一般都会选择直接使用 markdown 来编写。比如我们在之前用到的 go-daemon 库，功能比较简单，就是创建一个 daemon 进程，它的使用方式也比较简单，所以，作者就在项目 GitHub 地址的 README.md 中写所有信息的使用文档了。

而对于比较大型的项目，大多数项目都会自己开启一个官方网站，来说明项目的各种使用方法，比如 gorm 项目，这个库封装的函数多种多样，所以作者单独创建了一个 网站 来列出对 gorm 项目的说明。

使用网站来展示项目使用文档有一个额外工作，就是当版本升级的时候，要同时升级官方文档。否则的话，使用者就会经常遇到看旧文档、使用新框架的情况，而导致错误的用法。所以必须及时维护代码和文档的一致性。

那有没有办法将这两者结合一下呢？我们使用 markdown 编写说明文档，然后如果能自动将这些 markdown 文档转化为 HTML 网站，再将网站部署到服务器上，那就完美了。

确实是有这样的工具的，vuepress。vuepress 是一个 Vue 工具，基于 Vue 框架生成了一个 vuepress 的命令行工具，这个工具能将指定的 markdown 文件转化为 HTML 文件，而这个 HTML 文件是可以直接被使用者访问的。

vuepress 的编译需要两个目录，存放 markdown 的目录和生成 HTML 的目标目录。vuepress 是基于 Vue 的，正好 hade 框架也已经融合了 Vue。所以自然可以想到，直接将存放 markdown 的目录放在 hade 框架地址上，然后将生成 HTML 的目标目录定义为我们的 dist 目录。这样，就可以在 hade 框架上使用 npm 工具来生成 HTML 了。

编写 markdown

于是我们在根目录下创建 docs 目录，存放编写的 markdown 文件。

这里框架说明的 markdown 文件我都事先编写完成，也存放在 GitHub 上了，你可以比对查看。vuepress 的 markdown 文件格式，编写并没有什么特别，只需要你多阅读尝试，心里对哪种格式最终的页面展现是什么样子有理解就行。

比如想要首页的展示形式是这样：

它对应的 markdown 为 dos/README.md：

复制代码

```
---
home: true
actionText: 开始体验
actionLink: /guide/introduce
footer: MIT Licensed | Copyright © 2020-present jianfengye
features:
  - title: 基于协议
    details: 服务与服务间的协议是基于协议进行交互的。
  - title: 前后端协同
    details: 前后端协同开发
  - title: 命令行
    details: 有充分的命令行工具
  - title: 集成定时服务
    details: 如果你需要启动定时服务，提供命令进行定时服务的启动
  - title: 文档丰富
    details: 提供丰富的文档说明，提供丰富的文档说明
  - title: 开发模式
    details: 在开发模式下进行前后端开发，极大提高了开发效率和开发体验
```

根据 vuepress 的官方文档，我们需要在 docs 目录下创建一个.vuepress/config.js ，来给 vuepress 工具阅读，也就是用来告诉 vuepress 工具，你需要按照这个配置的信息生成 HTML 文件。这里的所有配置都在官网有说明。

我们的 config.js 的配置如下：

复制代码

```
module.exports = {
    title: "hade框架", // 设置网站标题
    description: "一个支持前后端开发的基于协议的框架", //描述
    dest: "./dist/", // 设置输出目录
    port: 2333, //端口
    base: "/v1.0/",
    head: [["link", {rel: "icon", href: "/assets/img/head.png"}]],
    themeConfig: {
        //主题配置
        // logo: "/assets/img/head.png",
        // 添加导航栏
        nav: [
            {text: "主页", link: "/"}, // 导航条
            {text: "使用文档", link: "/guide/"},
            {text: "服务提供者", link: "/provider/"},
            {
                text: "github",
                // 这里是下拉列表展现形式。
                items: [
```

```
                    {
                        text: "hade",
                        link: "https://github.com/gohade/hade",
                    },
                ],
            },
        ],
        // 为以下路由添加侧边栏
        sidebar: {
            "/guide/": [
                {
                    title: "指南",
                    collapsable: false,
                    children: [
                        "introduce",
                        "install",
                        "build",
                        "structure",
                        "app",
                        "env",
                        "dev",
                        "command",
                        "cron",
                        "middleware",
                        "swagger",
                        "provider",
                        "todo",
                    ],
                },
            ],
            "/provider/": [
                {
                    title: "服务提供者",
                    collapsable: false,
                    children: [
                        "app",
                        "env",
                        "config",
                        "log",
                    ],
                },
            ],
        },
    },
};

```

说明下几个重点配置项。

dest 这个配置项指定了我们生成 HTML 的目标文件夹，这里定义为 dist 目录。base 代表所有页面的访问前缀，我们定义为和版本号一致的 /v1.0/。这样设置最终会有什么效果呢？

由于 hade 会有多个版本，而我们希望访问地址的 URL 中带着版本信息，这样就能通过 URL 来访问不同的版本信息。比如 hade.funaio.cn/v1.0/，访问 v1.0 的版本信息；hade.funaio.cn/v1.1/ 访问 v1.1 的版本信息。所以使用 /v1.0/ 的 base，能让所有的访问信息都带上这个前缀，就能访问到不同版本的框架信息了。

themeConfig.nav 是配置导航栏的，我们的导航栏有四个信息。

所以在 themeConfig.nav 中需要定义四个子项，子项目的 text 表示显示文本，而 link 表示点击这个文本之后的链接地址。比如：

```
1 {text: "使用文档", link: "/guide/"},
```

复制代码

表示“使用文档”这个文本点击之后，会查找 /guide/ 目录下的 README 文件。

而 /guide/ 目录下的所有文件是通过 themeConfig.sidebar 设置侧边栏的，也就是刚才代码的第 29 到 47 行。最终通过这段的设置，点击首页的“使用文档”链接，就会进入如下的效果：

左边的每个链接都对应 docs/guide/ 中的每个 markdown 文件。

生成 HTML

docs 下的 markdown 文件都编写完成了，下面我们就安装 vuepress 并且生成 HTML。安装 vuepress，只需要使用 npm 命令：

```
1 npm install -D vuepress
```

复制代码

就能安装最新版本的 vuepress。截止 11 月 13 日，目前 vuepress 的 2.0 还处在 beta 版本，最稳定版本是 1.8.2。在安装过程中你可能会遇到这个错误：

```
1 TypeError: Cannot read property 'createHash' of undefined
```

复制代码

这个错误是提示我们的 Webpack 版本较低，使用命令升级 Webpack 就行。

```
1 npm i webpack@4.8.3
```

复制代码

安装好 vuepress 之后，我们就可以在 package.json 中设置 vuepress 的调试和编译命令了：

复制代码

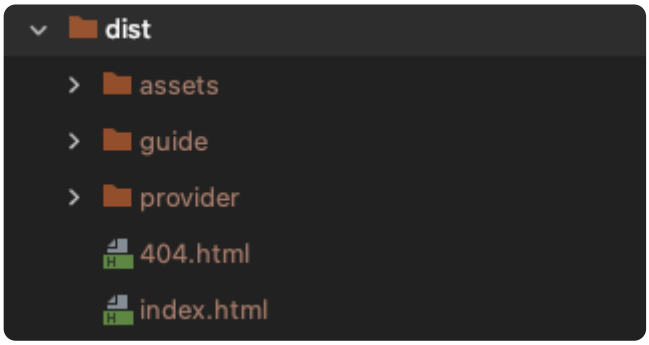
```
{
    "name": "hade",
    ...
    "scripts": {
        ...
        "docs:dev": "vuepress dev docs",
        "docs:build": "vuepress build docs"
    },
```

增加了 docs:dev 和 docs:build 来运行 vuepress。

这里我们使用 `npm run docs:build` ，就能生成对应 markdown 的 HTML 了，非常方便。

```
~/Documents/UGit/coredemo  geekbang/29  npm run docs:build

> hade@1.0.0 docs:build /Users/yejianfeng/Documents/UGit/coredemo
> vuepress build docs

wait Extracting site metadata...
tip Apply theme @vuepress/theme-default ...
tip Apply plugin container (i.e. "vuepress-plugin-container") ...
tip Apply plugin @vuepress/register-components (i.e. "@vuepress/plugin-register-components") ...
tip Apply plugin @vuepress/active-header-links (i.e. "@vuepress/plugin-active-header-links") ...
tip Apply plugin @vuepress/search (i.e. "@vuepress/plugin-search") ...
tip Apply plugin @vuepress/nprogress (i.e. "@vuepress/plugin-nprogress") ...

✔ Client
  Compiled successfully in 10.28s

✔ Server
  Compiled successfully in 6.84s

wait Rendering static HTML...
success Generated static files in dist.
```

可以看到 dist 目录中已经生成了对应的 HTML 文件：



最后将这个 HTML 文件部署到 Web 服务器中。这里我们部署在目标服务器的的 /webroot/hade_doc/dist_1.0 目录中。具体部署方法也没有什么难点，通过 FTP 或者 SETP 上传 dist 目录内容到目标目录即可。

我们的目标服务器配置的 Web 服务器为 Nginx，Nginx 如何配置，你可以参考：

复制代码

```
    server {
        server_name  hade.funaio.cn;
        access_log  logs/hade.access.log  main;
        error_log  logs/hade.error.log ;

        location /v1.0/ {
            alias /webroot/hade_doc/dist_1.0/;
            index  index.html index.htm;
        }

        location / {
            root   /webroot/hade_doc/dist_1.0/;
            index  index.html index.htm;
        }
    }
```

配置 hade.funaio.cn/v1.0/ 来访问目录路径 /webroot/hade_doc/dist_1.0/ ，同时 hade.funaio.cn/ 也是访问 /webroot/hade_doc/dist_1.0/。

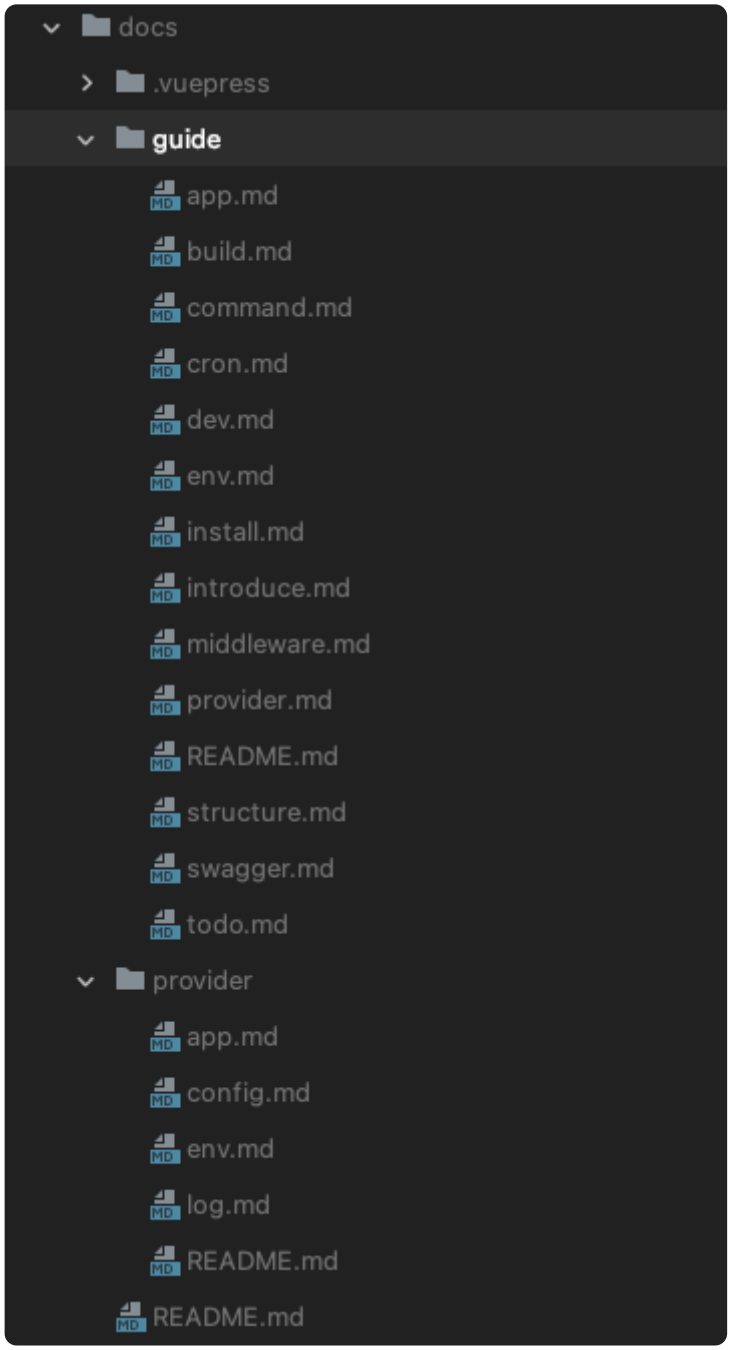
这样每次我们的版本有更新的时候，创建一个新的目标目录存放这个版本的 HTML，比如 /webroot/hade_doc/dist_1.1/。而在 nginx 上，只需要增加一个路径 /v1.1/ 来提供 hade.funaio.cn/v1.1/ 的访问路径，让使用者访问 v1.1 的文档。

于是，我们 hade 框架的网站文档：http://hade.funaio.cn 就正式搭建起来了。

看到这里，相信你应该理解了，我们的文档维护是基于 hade 项目中自带的 docs/ 目录下的 markdown 来进行的。你可以通过 GitHub 上的 markdown 来查看框架文档，也可以通过网站 hade.funaio.cn 来查看框架文档。这两者本质上都是通过 markdown 来编写的。

而 markdown 会随着 hade 框架的发布而发布，同时网站也是根据 markdown 生成的。这种方法既能避免文档和框架不一致的问题，又能大大降低维护网站的成本。

这节课我们没有修改框架的 Go 代码，主要创建了包含 markdown 文件的 docs 目录。目录截图放在这里供你你参考。所有代码也同步到了 geekbang/29 分支上了。



小结

今天我们为框架的升级和维护设计了一套完整的方案。hade 框架的发布和文档维护都有自己的独特设计，比如框架的发布，直接和 hade 框架的命令行工具进行了关联，只要发布了一个新版本，使用者就能直接用命令行工具使用这个新版本创建一个脚手架。而文档维护，是通过编写 markdown 以及 hade 框架已经集成的 Vue，来自动生成网站 HTML。

截止到这节课，hade 框架的功能就开发完毕了，GitHub 上的 coredemo 项目也就不再更新了，之前的所有代码，我们都会移动到真正的 hade 项目的 https://github.com/gohade/hade 地址，并且我们为代码发布了发布版本：v1.0.0，后续会基于这个版本不断迭代更新，如果你有兴趣，欢迎一起来完善这个框架。

思考题

马上我们会进入实战环节，使用 hade 框架开发具体应用，会使用到 Vue 和 Element-UI 的知识，当然在后面我们也会稍微介绍一些前端知识，不过如果你之前完全没有接触过，可以花一些时间预习一下，这样学习的时候才能事半功倍。

所以今天就给你分享一些相关资料：

- vue 中文官网，学习 Vue 必看内容
- element-ui 官网，一款快速搭建网站的 UI 框架，目前国内最火
- awesome-vue，收集 Vue 的一些优秀开源项目，绝对收藏
- vue router 官网，Vue 的路由必备
- webpack 官网，Vue 的打包工具

欢迎在留言区分享你的学习笔记。感谢你的收听，如果你觉得今天的内容对你有所帮助，也欢迎分享给你身边的朋友，邀请他一起学习。我们实战篇见。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 28 | SSH：如何生成发布系统让框架发布自动化？

下一篇 大咖助阵 | 大明：抽象，抽象，还是抽象

训练营推荐

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。